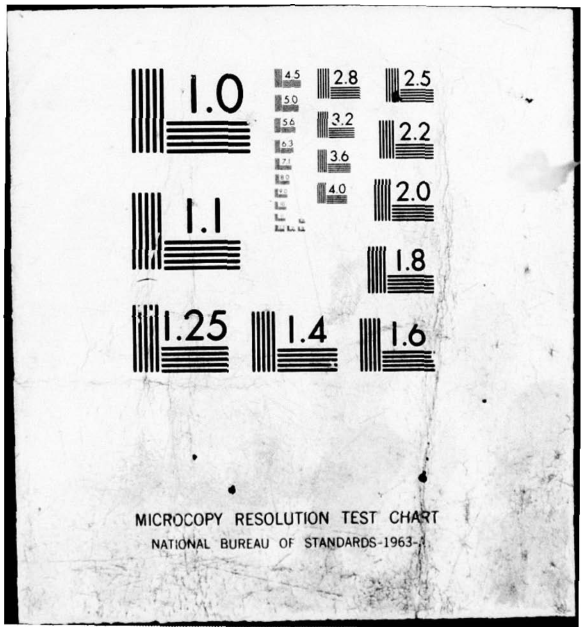
AD-A078 976 BDM CORP MCLEAN VA F/G 9/5
NET-2 NETWORK PROGRAM ANALYSIS. ADDENDUM TO USER'S MANUAL FOR R--ETC(U)
NOV 77 DAAG39-77-C-0150
UNCLASSIFIED BDM/W-77-573-TR NL

1 OF 1
AD
A078976

END
DATE
FILMED
2-80
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

475

1

THE BDM CORPORATION

LEVEL III

THE BDM CORPORATION

7915 Jones Branch Drive
McLean, Virginia 22101
Phone (703) 821-5000

11

30 NOV [redacted] 77

14

BDM/W-77-573-TR

12 37

6

NET-2 NETWORK PROGRAM ANALYSIS.

ADDENDUM TO USER'S MANUAL FOR RELEASE 9.1

This report was prepared for the Harry Diamond Laboratories under [redacted] DAAG 39-77-C-0150

15

391 962

18

FOREWORD

This report was prepared for the Harry Diamond Laboratories, Adelphi, Maryland, by The BDM Corporation under Contract DAAG39-77-C-0150. This document describes new features which have been incorporated into the IBM 370 version of the NET-2 Network Analysis Program, Release 9.1 under this contract.

~~TABLE OF~~ CONTENTS :

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Release 9 of the NET-2 Network Analysis Program was first made available in 1973 on IBM and CDC computers. It represented a significant improvement in capability over its predecessor, Release 8. A User's Manual was published in September 1973 which describes the features available in Release 9.

In the years since the Release 9 User's Manual was published there have been several additions to the capability of NET-2. This additional capability has been funded through contracts with the Naval Surface Weapons Center and the Harry Diamond Laboratories.

Two of the improvements resulted in major modifications to NET-2. These improvements are 1) the elimination of computational delay for basic circuit elements and the X variables, and 2) the addition of digital logic module modeling capability. These improvements have significantly extended the capability of NET-2.

Other improvements have been in the addition of new network elements. These new elements are the Gummel-Poon transistor model, a tabular s-plane transfer function, a dispersive transmission line, a power series V-I curve element, and a voltage controlled capacitance.

Collectively, these additions to the original Release 9 version have resulted in a new version of NET-2, called Release 9.1, available on both IBM and CDC computers. This report describes all of the new features in Release 9.1 and is intended as a supplement to the Release 9 User's Manual of September 1973.

CHAPTER II

ELIMINATION OF COMPUTATIONAL DELAY

In Release 9 and earlier releases of NET-2, a simultaneous solution is not obtained when element values are specified by mathematical expressions which depend on the network solution, for example:

```
R1      1      0      F1(N(1))
```

Instead, a computational delay is introduced. At each time step, the value expression is evaluated using network response variables from the previous time step. Then the network response for the current time step is evaluated. Simultaneities among auxiliary variables, or between auxiliary variables and element values, have not been allowed in NET-2. Such simultaneities have resulted in termination of the run with an error message to inform the user of the non-allowed condition.

Release 9.1 of NET-2 permits simultaneous solutions to be obtained for problems containing dependencies of the types described above. The circuit element values which may be included in simultaneous relationships are those for resistors, capacitors, inductors, voltage sources, and current sources. All allowed response variables for the above elements can be referenced by expressions which are part of the system of simultaneous equations.

Examples of equation systems which can now be handled simultaneously are shown in Figure II-1.

Under normal operation, Release 9.1 will handle all simultaneous relationships of the types shown in Figure II-1 in truly simultaneous fashion. Any simultaneities which are specified which are not of the type shown in Figure II-1 will be treated by introducing a computational delay. NET-2 will advise the user when this occurs. However, NET-2 will not permit simultaneities involving the AC small signal variables; the use of simultaneous relations involving these quantities will result in termination of the run.

1. Network solution simultaneous with respect to an element value

```
R1     1     0     F1(N(1))
```

2. Element values mutually dependent

```
R1     1     0     F1(R2)
R2     2     1     F2(R1)
```

3. Auxiliary variables mutually dependent

```
X1  = F1(X2)
X'2 = F2(X3)
X3  = F3(X'1)
```

4. Auxiliary variables, element values, and network solution mutually dependent

```
R1     1     0     F1(X1)
X1 = F2(N(1))
```

Figure II-1. Allowed Simultaneous Equation Systems

It is possible for the user to select other options for the handling of simultaneous relationships. If the following card

```
DEBUG  42
```

is included at indentation level 0 in the input deck, NET-2 will operate in the mode of Release 9 and always insert computational delays. If the card

```
DEBUG  43
```

is included in the input deck, NET-2 will terminate execution if any computational delays are required. Table II-1 indicates the action of NET-2 for the various combinations of DEBUG 42 and DEBUG 43 cards.

TABLE II-1. NET-2 ACTION FOR COMPUTATIONAL DELAY OPTIONS

	DEBUG 43 INCLUDED	DEBUG 43 MISSING
DEBUG 42 INCLUDED	ABORT IF ANY SIMULTANEITIES OCCUR	HANDLE ALL SIMULTANEITIES AS COMPUTATIONAL DELAYS
DEBUG 42 MISSING	ABORT IF NON-ALLOWED RELEASE 9.1 SIMULTANEITIES OCCUR	ALL ALLOWED RELEASE 9.1 SIMULTANEITIES HANDLED SIMULTANEOUSLY. NON-ALLOWED SIMULTANEITIES HANDLED AS COMPUTATIONAL DELAYS.

CHAPTER III

DIGITAL LOGIC MODULES

A. LOGIC MODULES

Digital logic arrays may be described to NET-2 by defining and referencing one or more logic modules. The logic module is defined by a format very similar to that used for subnetwork definition in NET-2. The primary distinction is the use of the keyword LOGIC in the initial line of the description. The logic module definition is then expressed in terms of logic elements and references to other logic modules. The logic module may not contain any non-logic NET-2 elements or references to NET-2 subnetworks.

The logic module definition is best described by means of an example:

```
DEFINE  LOGIC  COUNTER  A  B/1  2  3
```

The keyword DEFINE specifies that either a logic module or a subnetwork is being defined; the second keyword LOGIC clarifies that it is indeed a logic module. The third item is the name of the logic module, composed only of alphabetic characters. This is followed by a listing of the interface nodes for the module, with the output nodes appearing first, followed by the input nodes. A slash is used to separate the output nodes from the input nodes. Thus, in the above example, logic module COUNTER is being defined with output nodes A and B, and input nodes 1, 2, and 3. A logic module is restricted to a maximum of 47 output nodes.

The logic module definition must appear at indentation level 0 in the NET-2 input description. Subsequent lines of the definition list the logic elements and logic module references which constitute the logic module being defined. These elements and references are written starting at indentation level 1, using the interface nodes and other arbitrarily chosen node names for internal connection purposes. Compound node names (e.g., ABC1.XY) may not be used for connection purposes inside of a logic module. Any arbitrary alphameric name may be used for node names. Node 0 has no special significance in the logic module.

It is possible to store a logic module in the Stored Model Library, with retrieval at a later time for inclusion in a network simulation. This is done by replacing the keyword DEFINE with the keyword MODEL, just as for ordinary subnetworks. Other than the keyword change, the logic module definition is identical. For example:

```
MODEL  LOGIC  COUNTER  A  B/1  2  3
```

Logic modules which have been stored in the Stored Model Library can be listed and changed in the same manner as subnetworks.

B. LOGIC ELEMENTS

Logic modules are defined in terms of logic elements. These logic elements include AND, NAND, OR, NOR, NOT, Exclusive OR, RST flipflop, time delay, and other logic modules. The description of each kind of logic element is given below.

1. AND Element

The AND element accomplishes the function of logical AND. It has one output and as many inputs as desired. The format is:

ANDn OUT IN1 IN2 .....

where: ANDn = element ID

OUT = output node name

IN1, IN2, etc. = input node names

2. NAND Element

The NAND element is logically identical to an AND element with a complemented output. It has one output and as many inputs as desired. The format is:

NANDn OUT IN1 IN2 .....

where: NANDn = element ID

OUT = output node name

IN1, IN2, etc. = input node names

3. OR Element

The OR element accomplishes the function of logical OR. It has one output and as many inputs as desired. The format is:

ORn OUT IN1 IN2 .....

where: ORn = element ID

OUT = output node name

IN1, IN2, etc. = input node names

4. NOR Element

The NOR element is logically identical to an OR element with a complemented output. It has one output and as many inputs as desired. The format is:

NORn OUT IN1 IN2 .....

where: NORn = element ID

OUT = output node name

IN1, IN2, etc. = input node names

5. NOT Element

This element performs logical complementation. The value at the output node is the logical complement of the value at the input node. There is a single output node and a single input node. The format is:

NOTn OUT IN

where: NOTn = element ID

OUT = output node name

IN = input node name

6. Exclusive OR Element

This element implements the Exclusive OR function. It has one output and as many inputs as desired. The format is:

EORn OUT IN1 IN2 .....

where: EORn = element ID

OUT = output node name

IN1, IN2, etc. = input node names

The output value = 1 if there is one and only one input node with a 1 value present. Otherwise, the output value = 0.

C

7. RST Flipflop

This element is a logical model of an RST flipflop. It has one output and three input nodes. The format is:

RSTFFn OUT R S T

where: RSTFFn = element ID

OUT = output node name

R = reset input node name

S = set input node name

T = trigger input node name

The RST flipflop output value depends upon the previous time value of the output Fo, the previous time value of the trigger To, and the current values of the reset input R, the set input S, and the trigger T. In Boolean notation the logical characteristic of the RST flipflop is:

$$F = \overline{R} * (S + (T * \overline{To}) \veebar Fo)$$

where $\overline{R}$ represents the logical complement of R, * denotes logical AND, + denotes logical OR, and ⊻ denotes Exclusive OR.

This relationship can be expressed in Karnaugh map form as follows:

	R	0	0	0	0	1	1	1	1
	S	0	0	1	1	0	0	1	1
	T	0	1	0	1	0	1	0	1
To	Fo								
0	0	0	1	1	1	0	0	0	0
0	1	1	0	1	1	0	0	0	0
1	0	0	0	1	1	0	0	0	0
1	1	1	1	1	1	0	0	0	0

From the above description it is seen that the R input has the highest precedence, followed by the S input, with the T input having the lowest precedence. When both the R and S inputs are 0, the output state changes only when the T input transitions from 0 to 1.

8. Time Delay Element

The time delay element introduces a time delay between the input node and the output node. Three different kinds of time delay are available.

The first kind of time delay is the computational delay. The format is:

DELAYn OUT IN

where: DELAYn = element ID

OUT = output node name

IN = input node name

This element provides a time delay of zero time between the input and output nodes. The delay runout always occurs instantaneously, within the current computational time step. If a transition occurs at the input of a computational delay element, an entry is made into a special first-in, first-out queue. As this queue is emptied, the output of the appropriate delay is complemented, reflecting the input transition, and the state of each element affected by the transition is recomputed.

The second kind of delay time assumes that the time delay value is identical for both 0 to 1 transitions and 1 to 0 transitions at the input node. The format is:

DELAYn OUT IN Value

where: DELAYn = element ID

OUT = output node name

IN = input node name

Value = delay time value

The delay time value must be a numerical constant.

The third kind of delay element assumes that the delay time is different for 0 to 1 transitions and 1 to 0 transitions at the input node. The format is:

DELAYn OUT IN Value1 Value2

where: DELAYn = element ID

OUT = output node name

IN = input node name

Value1 = delay time for 0 to 1 transition at the input node

Value2 = delay time for 1 to 0 transition at the input node

The delay times must be numerical constants.

9. Logic Module Reference

A logic module which has been defined in the NET-2 input or whose definition appears in the Stored Model Library may be referenced within another logic module. Thus the logic module reference can be thought of as the specification of a specialized logic element. The format for the reference will be illustrated by a reference to the COUNTER logic module previously discussed:

```
COUNTER35  23  56  Q  3  5R
```

The element ID is formed by appending a suffix to the logic module name. This suffix may be any alphameric sequence provided it begins with a numeric character. The element ID is followed by a listing of the interface nodes to which the element is connected, using the same ordering as was used in the logic module definition. Note that the slash separator between output and input interface nodes is not used for logic module references.

C. LOGIC INTERFACE

The logic modules provide a means of describing a logical system of any complexity. However, a means must be provided for interfacing one or more logic modules to the NET-2 network description. This is accomplished by a NET-2 network element, the logic interface element. It interfaces a specified logic module to the network, and accomplishes the translation between network node values and the 0, 1 space of the logic module.

The logic interface element is very similar in function to a subnetwork reference element. It specifies the name of the logic module, the network nodes which interface with it, and provides optional information on translation which must occur for nodal values across the interface boundary. The format is best illustrated by using the COUNTER logic module example presented earlier. A typical interface of the COUNTER logic module with the NET-2 circuit/system network is given by:

```
COUNTER29  4  6  2  58  B
```

This has the exact appearance of a logic module reference, with an element ID followed by the network node names to which the logic module is connected.

All logic interfaces must appear at the main circuit level, at indentation level 0. Logic modules may not be referenced from the network from inside subnetworks.

The above example does not explicitly specify the translation which must occur between network node values and logic node values. The user may specify translation for individual nodes, as in the example:

```
COUNTER45  4  6(-2, 3.1)  2  58(3.14, 1)  B
```

Here we have specified a translation number pair, always enclosed by parentheses and always following the node name without intervening spaces, for two of the nodes. Let this number pair be represented by (a,b). For output nodes a = the network node value for a logical 0, and b = the network node value for a logical 1. Thus, for output node 6 in the example, a logical 0 in the corresponding logic output node for logic module COUNTER would appear as a network node value of 3.1.

For input nodes, b = the logical value to be assumed by the node whenever the network node value is greater than a; otherwise, the logical complement of b is used as the logical value. Thus, for the example, if network node 58 (a logic module input node) has a value greater than 3.14, the corresponding logic value for that node is 1; if the network node value is equal to or less than 3.14, then the corresponding logic value is 0.

Default values apply to any node for which the number pair is not specified. The default values are a = 0, b = 1. Thus, if the default values are used, the translation which occurs at the network-logic module interface is identical to that used by the network logic elements previously available in NET-2.

The values for a and b must be expressed as numerical constants. Mathematical expressions may not be used.

D. NODE INITIALIZATION

The user may specify initial values for both network and logic module nodes through the INITIAL entry. An example is:

```
INITIAL
    N(23)=12
    N(PQ6.3)=-3.5
    L(COUNTER45.7)=0
    L(ABC9.XYZ7.Q)=1
```

The INITIAL Entry appears at indentation level 0, with subsequent lines, one for each node to be initialized, appearing at indentation level 1. Nested node names may be used, and all node names are written as referenced from the main circuit level. The form N(x) specifies the initial value for network node x. The form L(x) specifies the initial value for logic node x (the value is restricted to 0 or 1). Since logic nodes can be reached only through a logic interface element, it is apparent that all logic node names will be compound names, specifying first the logic module accessed directly by the logic interface, followed by any descent through nested logic modules required to reach the module of interest, and concluding with the node name within the desired logic module.

The initial value assigned to any node will be maintained for all steady state solutions and as the initial condition for any transient solutions.

E. NODE REFERENCES

The value of a logic module node may be referenced for purposes of printing output or for use in a mathematical expression in NET-2. This is accomplished by using the notation L(x), where x is the logic node name, described previously for node initialization. Thus, for example:

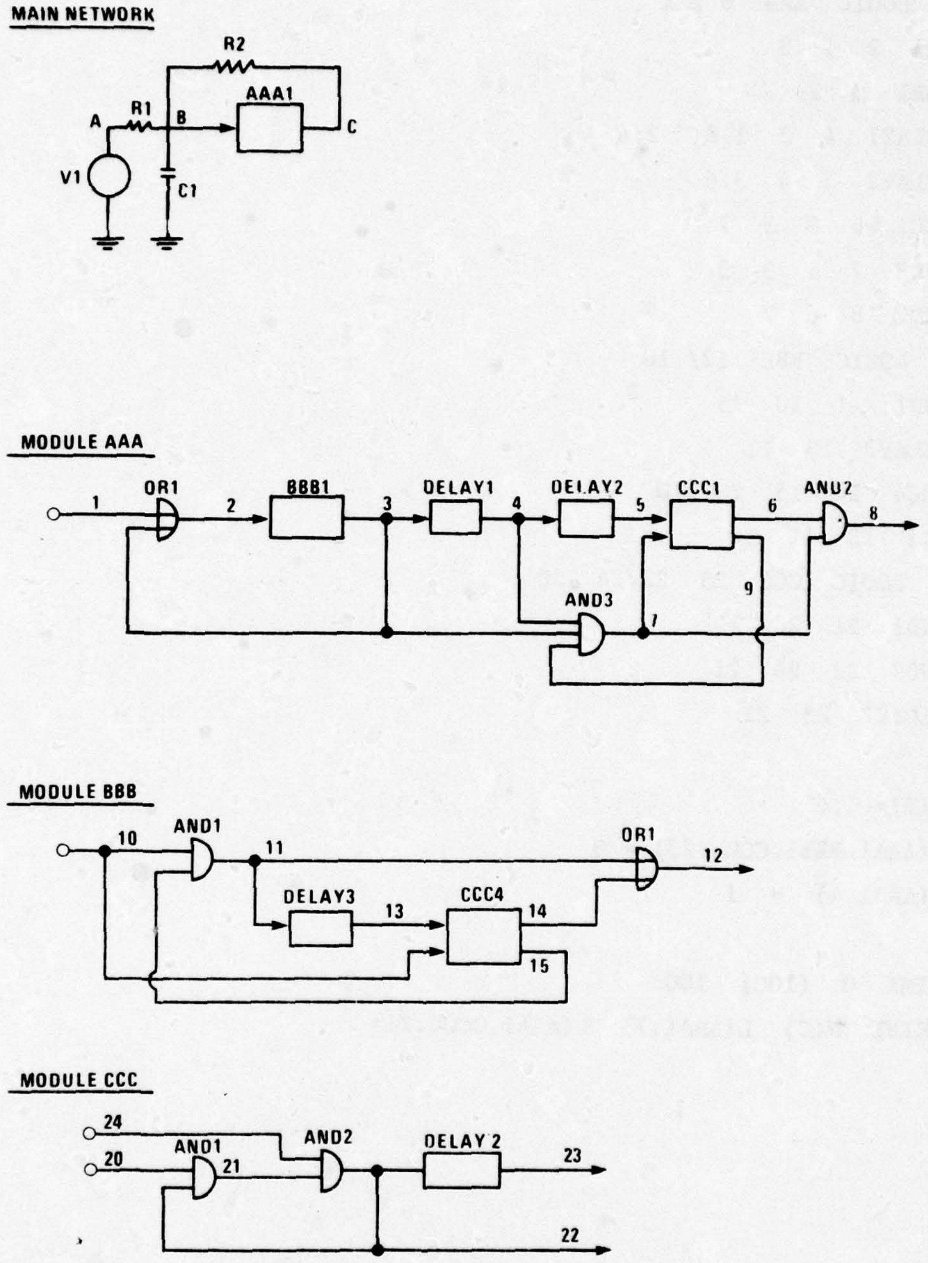
```
STATE1
     PRINT L(COUNTER45.7)  N(23)
```

will print the values of logic node COUNTER45.7 and network node 23 as a function of time. Obviously, the logic node values are restricted to 0 and 1.

F. LOGIC EXAMPLE

An example logic problem is illustrated in Figure III-1. This example is not chosen to represent any particular meaningful logic situation, but rather to illustrate the manner in which the logic modules and other language features are described to NET-2. A NET-2 input description corresponding to the example in the figure is:

```
*LOGIC EXAMPLE

V1  A  0  .35

R1  A  B  12

R2  B  C  39

C1  B  0  2.6

AAA1  C(-3, 1)  B
```



MAIN NETWORK

MODULE AAA

MODULE BBB

MODULE CCC

Figure III-1. Logic Example

```
DEFINE  LOGIC  AAA  8 / 1
     OR1  2  1  3
     BBB1  3  2
     DELAY1  4  3  1.8,  2.4
     DELAY2  5  4  3.6
     CCC1  6  9  5  7
     AND3  7  4  3  9
     AND2  8  6  7
DEFINE  LOGIC  BBB  12/ 10
     AND1  11  10  15
     DELAY3  13  11
     CCC4  14  15  13  10
     OR1  12  11  14
DEFINE  LOGIC  CCC  23  22/24  20
     AND1  21  20  22
     AND2  22  24  21
     DELAY2  23  22
INITIAL
     N(B)=-1.6
     L(AAA1.BBB1.CCC4.23) = 0
     L(AAA1.4)  =  1
STATE1
     TIME  0  (100)  100
     PRINT  N(C)  L(AAA1.7)  L(AAA1.CCC1.21)
END
```

CHAPTER IV

GUMMEL-POON BIPOLAR TRANSISTOR

A. CIRCUIT DESCRIPTION

The Gummel-Poon model[1] for the bipolar transistor is available in NET-2. In the simplest form, this model reduces to the basic Ebers-Moll equations. With proper parameter selection, the model includes the effects of conductivity modulation, base push-out, Early effect, impact ionization, and bias dependence of the current gain. The model in its present form does not contain nuclear radiation effects and the data reduction feature for device parameters is not available.

The same model is used for both PNP and NPN devices. Specification of PNP or NPN is made by setting a device parameter.

The Gummel-Poon transistor is a modeled device and requires device parameters in the Device Parameter Library. The model number is 10.

The format for the Gummel-Poon transistor is:

Tn (p) E B C Type Mode

where: Tn = transistor ID
p = optional parallel segment designation
E = emitter node name
B = base node name
C = collector node name
Type = device type name
Mode = optional mode designation

The optional mode designation may be used to assist NET-2 in obtaining the correct DC steady state solution when a Gummel-Poon transistor is involved in a bistable circuit configuration. The word OFF is the only legal mode designation available for this element; it is used to instruct NET-2 to maintain the specified transistor in the cutoff condition during the DC steady state solution. The cutoff constraint is automatically removed during the transient solution. The mode designation may be superseded in a specific State, Monte Carlo, or Optimization calculation. Blind use of the word OFF in unnecessary situations can lead to serious computational errors since transistor cutoff is always enforced by NET-2 in such situations.

[1] H. C. Poon, "Modeling of Bipolar Transistor Using Integral Charge-Control Model with Application to Third-Order Distortion Studies," IEEE Transactions on Electron Devices, ED-19, no. 6, pp 719-731, June, 1972.

B. EQUIVALENT CIRCUIT

The equivalent circuit for the Gummer-Poon model is shown in Figure IV-1. The dominant component of the collector current is given by the difference of the injection currents.

$$I_{cc} = I_r - I_f$$

where $I_f = (I_s Q_{bo}/Q_b) \left(e^{\theta v_1} -1 \right)$

and $I_r = (I_s Q_{bo}/Q_b) \left(e^{\theta v_2} -1 \right)$

The base charge Q_b is a function of the junction voltages, currents and capacitances.

$$Q_b = Q_{bo} + C_{te} v_1 + C_{tc} v_2 + B \tau_f I_f + \tau_r I_r$$

$$C_{te} = C_e \left(1 - \min \left(\frac{v_1}{V_{ze}}, .9 \right) \right)^{-N_e}$$

$$C_{tc} = C_c \left(1 - \min \left(\frac{v_2}{V_{zc}}, .9 \right) \right)^{-N_c}$$

The base push-out function B is given by

$$B = 1 + B_1 [1 - B_2 v_2] - (I_c B_3 / v_2) [1 - B_2 v_2]^2$$

The collector current I_c is a positive quantity and the junction voltage v_2 is negative for a device operating in the active normal region.

The re-combination currents I_{bc} and I_{be} are given by

$$I_{be} = I_1\left(e^{\theta v_1}-1\right) + I_2\left(e^{\theta v_1/m_e}-1\right)$$

$$I_{bc} = I_3\left(e^{\theta v_2/m_c}-1\right)$$

The impact ionization current I_A is given by[1]

$$I_A = (2\alpha_n I_c/b_n)(V_{zc} + v_2)[1 + \eta E_o(1 - v_2/v_{zc})^{-1/2}]e^{-b_n/E_m}$$

$$E_m = \left\{(2q_e N_o V_{zc}/\epsilon)(1 - v_2/V_{vc})^{1/2}[1 - I_c/(q_e V_s A_e N_o)]\right\}^{1/2}$$

$$-E_o[1 - v_2/V_{zc}]^{1/4}\left[1 - I_c/(q_e V_s A_e N_o)\right]^{n_A}$$

$$\eta = 3/4(\pi b_n)^{1/2}\left[\epsilon/(2q_e N_o V_{zc})\right]^{3/4}$$

If ϵ is given a value of zero, the impact ionization calculation will be omitted, with a resultant reduction in computation time.

The inactive collector capacitance is calculated as

$$C_i = r_A C_c\left(1 - \min\left(\frac{v_3}{V_{zc}}, .9\right)\right)^{-N_c}$$

and the diffusion capacitances are

$$C_{de} = B\ \tau_f\ \partial I_f/\partial v_1$$

and

$$C_{dc} = \tau_r\ \partial I_r/\partial v_2$$

[1] H.C. Poon and J.C. Meckwood, "Modeling of Avalanche Effect in Integral Charge Control Model," IEEE Transactions on electron Devices, ED-19, no. 1, January, 1972.

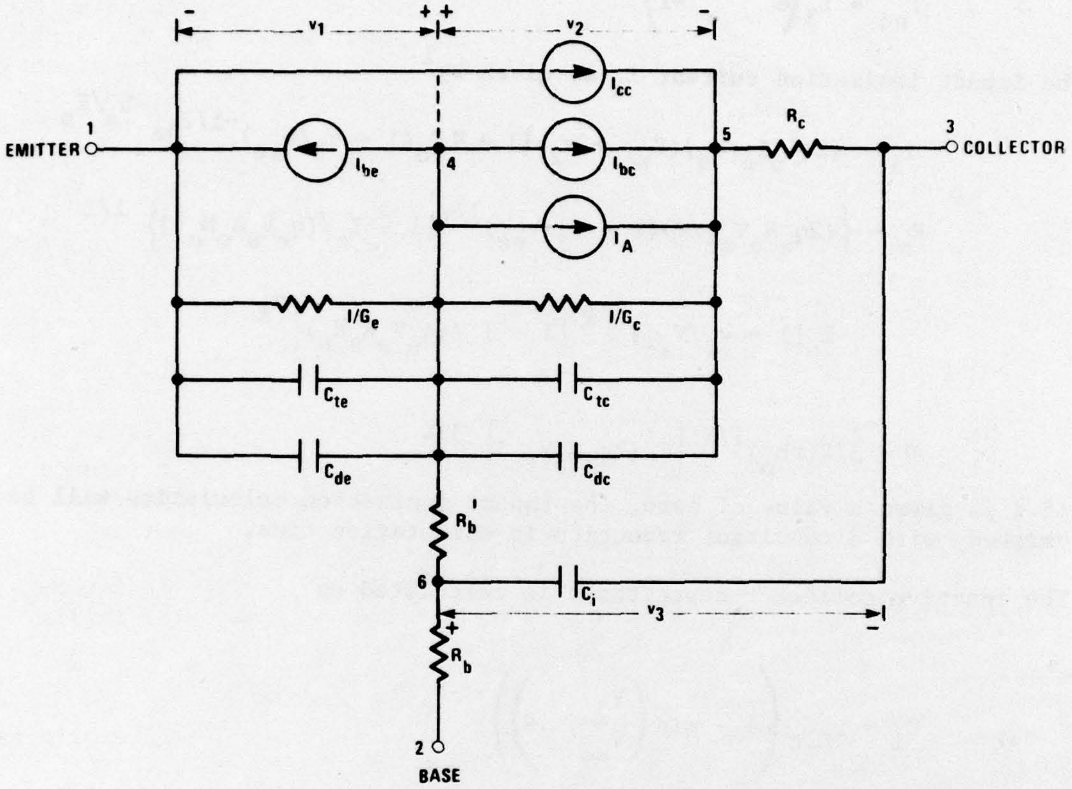


Figure IV-1. Gummel-Poon Transistor Equivalent Circuit

The voltage-dependent resistance R_b' is given by

$$R_b' = R_{BP} Q_{bo} / Q_b$$

The equations and equivalent circuit shown above refer to polarities for the NPN device. However, identical equations may be used for a PNP device if the polarities of v_1, v_2, and v_3 are reversed, and the directions of current flow for I_{cc}, I_{bc}, I_{be}, and I_A are reversed on the equivalent circuit. This reversal is done automatically by NET-2 so that the user needs only to specify whether an NPN or PNP device is desired by appropriately setting the polarity parameter S in the device library. The signs of all parameter values except S remain unchanged when going from NPN to PNP. However, the values for α_n, b_n, and V_s will be different for a PNP device, because these values will then refer to holes rather than electrons.

C. GUMMEL-POON TRANSISTOR REFERENCES

The symbols used by NET to represent the Gummel-Poon transistor device parameters are given in Table IV-1. The user may control and reference any quantity for which a NET symbol has been assigned. Only numerical constants may be used for parameter values.

Reference to a device parameter within the LIBRARY Entry is accomplished by simply using the symbolic parameter name. References in all other entries are made using the symbolic name Tn.x where x is the parameter name.

The mode designation for a transistor may be altered from its original status using the name Tn.MODE followed by the word OFF or NONE. OFF establishes the cutoff mode for the transistor during the DC steady state calculation; NONE removes a previously established OFF mode.

Internal nodes 4, 5, and 6 in the bipolar transistor model are available for connection purposes using Tn.y as the node name, where y is 4, 5, or 6. The node voltage may be referenced by using N(Tn.y) as the symbolic name.

The user may reference the base-emitter voltage, the base-collector voltage, the collector-emitter voltage, the emitter current, the base current, the collector current, and the total transistor power dissipation by using the symbolic names VBE(Tn), VBC(Tn), VCE(Tn), IE(Tn), IB(Tn), IC(Tn), and P(Tn), respectively.

TABLE IV-1. GUMMEL-POOL BIPOLAR TRANSISTOR SYMBOLS (MODEL 10)

EQUATION SYMBOL	NAME	NET SYMBOL	NET UNITS	COMMENTS
A_e	EMITTER AREA	AE	cm	
α_n	AVALANCHE COEFFICIENT	ALPHA	cm	REFERS TO ELECTRONS FOR NPN, HOLES FOR PNP
b_n	CRITICAL FIELD	BN	v/cm	REFERS TO ELECTRONS FOR NPN, HOLES FOR PNP
B_1	BASE PUSH-OUT COEFFICIENT	B1	NONE	
B_2	BASE PUSH-OUT COEFFICIENT	B2	V^{-1}	
B_3	BASE PUSH-OUT COEFFICIENT	B3	v/ma	
C_c	COLLECTOR-BASE TRANSITION CAPACITANCE AT $v_2 = 0$	CC	pf	
C_e	EMITTER-BASE TRANSITION CAPACITANCE AT $v_1 = 0$	CE	pf	
ϵ	DIELECTRIC CONSTANT OF COLLECTOR REGION	EPS	pf/cm	
E_o	AVALANCHE FIELD COEFFICIENT	EO	v/cm	
G_c	COLLECTOR-BASE OHMIC LEAKAGE ADMITTANCE	GC	mmho	
G_e	EMITTER-BASE OHMIC LEAKAGE ADMITTANCE	GE	mmho	
I_s	COEFFICIENT FOR DOMINANT COMPONENT OF COLLECTOR CURRENT	IS	ma	ALWAYS POSITIVE
I_1	IDEAL BASE CURRENT COEFFICIENT	I1	ma	ALWAYS POSITIVE
I_2	NON-IDEAL BASE CURRENT COEFFICIENT	I2	ma	ALWAYS POSITIVE
I_3	REVERSE BASE CURRENT COEFFICIENT	I3	ma	ALWAYS POSITIVE
m_c	REVERSE BASE CURRENT EMISSION COEFFICIENT	MC	NONE	
m_e	FORWARD BASE CURRENT EMISSION COEFFICIENT	ME	NONE	
n_A	MOBILE CARRIERS EXPONENT	NA	NONE	
N_c	COLLECTOR-BASE GRADING COEFFICIENT	NC	NONE	
N_e	EMITTER-BASE GRADING COEFFICIENT	NE	NONE	
N_o	COLLECTOR DOPING COEFFICIENT	NO	cm^{-3}	
Q_{bo}	BASE CHARGE AT ZERO BIAS	QBO	pc	ALWAYS POSITIVE
q_e	ELECTRONIC CHARGE MAGNITUDE	QE	pc	ALWAYS POSITIVE

TABLE IV-1. GUMMEL-POOL BIPOLAR TRANSISTOR SYMBOLS (MODEL 10) (CONTINUED)

EQUATION SYMBOL	NAME	NET SYMBOL	NET UNITS	COMMENTS
r_A	RATIO OF INACTIVE BASE AREA TO ACTIVE BASE AREA	RA	NONE	
R_b	OUTER BASE RESISTANCE	RB	Kohm	
R_{BP}	INNER BASE RESISTANCE COEFFICIENT	RBP	Kohm	
R_c	COLLECTOR BULK RESISTANCE	RC	Kohm	
NONE	DEVICE POLARITY CONSTANT	S	NONE	+1 FOR NPN, −1 FOR PNP
Θ	CURRENT GENERATOR CONSTANT	TH	v^{-1}	$[q_e/kT]$
τ_f	FORWARD BASE TRANSIT TIME	TF	nsec	
τ_r	REVERSE BASE TRANSIT TIME	TR	nsec	
V_s	SATURATION VELOCITY	VS	cm/nsec	REFERS TO ELECTRONS FOR NPN, HOLES FOR PNP
V_{zc}	COLLECTOR-BASE CONTACT POTENTIAL	VZC	v	
V_{ze}	EMITTER-BASE CONTACT POTENTIAL	VZE	v	

CHAPTER V

TABULAR S-PLANE TRANSFER FUNCTION

The XFTAB element is a system element which performs time and frequency domain transformations on an input signal using an empirical transfer function. The transfer function is tabulated as amplitude and phase data as a function of frequency.

NET-2 uses the tabular frequency data directly in the frequency domain. In the time domain NET constructs a table of the time domain response of the element to a unit step input; it then uses superposition techniques to calculate the time domain response to an arbitrary input, viewing the arbitrary input as a time series of infinitesimal step functions of varying amplitudes. Operation in both the time and frequency domain is linear and simultaneous.

The element is described by the following format:

```
XFTABn   IN   OUT   Nbuf   TABLEf   TABLEt
```

```
where XFTABn = element ID
      IN     = input node
      OUT    = output node
      Nbuf   = size of time history buffer
      TABLEf = name of table containing amplitude and phase data
      TABLEt = name of table containing unit step response data
```

TABLEf is a two-dimensional table which is described elsewhere in the NET-2 input. The table is of dimension Nfreq x 2 where Nfreq is the number of frequency points in the table. In constructing the table the normal NET-2 formatting for two-dimensional tables is used. However, the information in the table is unique to the XFTAB element and the table must not be referenced except by an XFTAB element.

An example of the specification for TABLEf is:

```
TABLE23  2
2  .001
.5  .67  -48
1.  .85  -15
2. 1.3   17
5.  .67  28
```

The first line of the table lists the table name and the number of dependent variable values to be listed for each independent variable value. Since there are two entries (amplitude and phase) for each frequency value, the number of columns is always 2.

The second line of the table is used to convey information about the amplitude values. The second line consists of two values, an amplitude type code and an amplitude reference value (A_{ref}). Amplitude data (A) may be given using linear, logarithmic, or decibel values. If linear amplitude information is given, the amplitude type code = 0 and the true amplitude is given by:

$$A_{true} = A + A_{ref}$$

If logarithmic amplitude information is given, the amplitude type code = 1 and the true amplitude is given by:

$$A_{true} = 10^{A} + A_{ref}$$

If decibel amplitude information is given, the amplitude type code = 2 and the true amplitude is given by:

$$A_{true} = 10^{(A/20)} + A_{ref}$$

A series of lines, beginning with the third line, list the frequency data in order of ascending frequency. Each line lists, in order, the values of frequency, amplitude, and phase (in degrees) for each data point in the frequency domain for the transfer function. Linear interpolation is used between successive data points by NET. Sufficient data must be given to permit NET to correctly calculate the time domain response which corresponds to the frequency domain. Zero frequency must not be specified.

TABLEt is a one-dimensional table, called the unit step response table. It is used to contain the time domain response which occurs when the XFTAB element is subjected to a unit step on its input. The unit step is assumed to be applied at t = 0 with an excursion from 0 to 1. NET-2 automatically calculates the unit step response at times specified by the user, using the frequency data in TABLEf to make the calculation. Thus the user must specify the values of time at which the unit step response is to be calculated. These time values are entered in ascending order as the independent variable values in the TABLEt definition. The first time value specified corresponds to t = 0; the last time value specified is the time at which the unit step response can be considered to have reached the steady state.

The time values between the first and last should be chosen judiciously to permit NET to calculate a reasonable approximation of the waveform corresponding to the unit step response. NET will use straight line interpolation between successive time points in the unit step response table when calculating the response of the element to an arbitrary input.

The user must specify the dependent variable value for the last time point in the unit step response table. This value will be the final steady state value for the unit step response (i.e., the output node value of the element after an input node value of 1 has been applied for an infinitely long time). A value of 0 may be used for all other dependent variable values in TABLEt which are not specified by the user. NET-2 will calculate values for all of these intermediate dependent variable values using the frequency data.

An example of a TABLEt definition is:

```
TABLE56

0  0
1  0
2  0
5  0
8  .75
```

In order to calculate the time domain response to an arbitrary input NET requires the most recent past history of the input waveform for a period of time equal to the time span defined in TABLEt. This past history is maintained in an internal buffer as a set of ordered input waveform points, using linear interpolation between successive points. The number of points which may be contained in the buffer at any one time is specified by the Nbuf parameter. The buffer will attempt to contain as detailed a time history as possible within the size constraints of the buffer. If too many points are introduced into the buffer, an editing algorithm will identify those points which do not significantly contribute to the waveform shape and delete them. If, however, only significant points remain in the buffer so that no deletion can occur, and an attempt is made to add additional significant points while the buffer is filled to capacity, a diagnostic message will be printed and the run terminated. The user will be required to specify a larger buffer size in order to successfully complete the run. The editing algorithm which is used is similar to that used for the DELAY and TLINE elements.

CHAPTER VI

DISPERSIVE TRANSMISSION LINE

The dispersive transmission line element is a circuit element which models the behavior of transmission lines containing nonlinear equivalent circuit elements. The transmission line is modeled by NET-2 as a series of cascaded sections. Figure VI-1 shows a typical lumped section, modeled in terms of electrical circuit elements. Each section consists of a series inductance L, a series resistance R, a shunt capacitance C, and a shunt conductance G.

The user may assign empirical dependencies for the values of L, R, C, and G in a typical section. The empirical dependencies are specified by one-dimensional table references where the series element values are a function of element current, and the shunt element values are a function of the element voltage.

The actual transmission line model consists of a series of these sections cascaded together with three terminal nodes as shown in Figure VI-2. The user specifies the number of sections to be cascaded to form a particular transmission line. Only three nodes are introduced into the network solution of each transmission line. However, solution time and core storage requirements depend upon the number of transmission lines, the number of sections in a given transmission line, and the number of straight line segments in the corresponding table definitions. The transmission line model does not generate computational delay.

The format for the dispersive transmission line element is:

DTLINEn (p) a b c m Ltable Ctable Rtable Gtable Lvalue, Cvalue, Rvalue, Gvalue

where DTLINEn = element ID
p = optional parallel segment designation
a = node name for terminal node a
b = node name for terminal node b
c = node name for terminal node c
m = number of sections in line
Ltable = table ID for dependence of L on current
Ctable = table ID for dependence of C on voltage
Rtable = table ID for dependence of R on current
Gtable = table ID for dependence of G on voltage
Lvalue = coefficient for value of L
Cvalue = coefficient for value of C
Rvalue = coefficient for value of R
Gvalue = coefficient for value of G

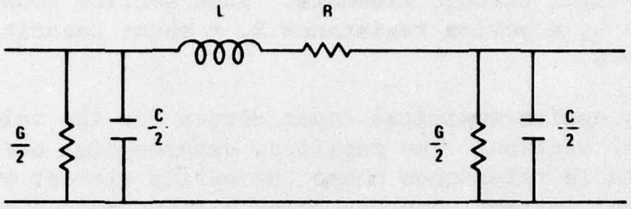


Figure VI-1. Typical Transmission Line Section

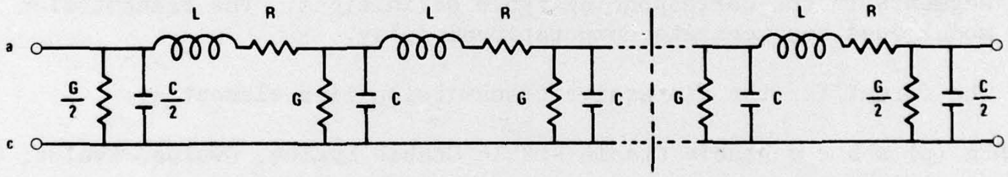


Figure VI-2. Transmission Line Model

The value of an element used in a particular section is calculated as the product of the element table value and its coefficient value. Thus, for example, L = Lvalue x Ltable, where the value for Ltable is an empirical function of the current through the L element. Lvalue, Cvalue, Rvalue, and Gvalue may be represented by numerical constants or by mathematical expressions. Note that the form of the mathematical expression may introduce computational delay into the calculation. Values for Lvalue, Cvalue, Rvalue, and Gvalue may be omitted; NET-2 will supply a default value of 1 for omitted values. The series resistance value may not assume a value of 0 during the time domain steady state solution or the frequency domain solution for zero frequency.

The IDs for Ltable, Ctable, Rtable, and Gtable are given without any argument list. Data in the tables is given only non-negative values of the independent variable (current or voltage). NET-2 assumes that the table values are mirrored for negative values of the independent variable. Thus, if the value of 10 is listed for L when the current through L is 1, NET-2 will also use a value of 10 for L when the current through L is -1.

If constant values are desired for an equivalent circuit element, a table with a single entry of unity may be used along with a scaling factor specified by the corresponding coefficient value. A table with unit value can be specified by:

```
TABLE1

   0  1
```

which gives a value of 1 regardless of the independent variable value.

The DTLINE element may not be used to calculate complex (nonzero) frequency domain response during the transient response calculation (TIME>0). All other uses of the element are permitted.

An example of the use of the DTLINE element is given by

```
DTLINE35  4  6  9  50 TABLE2  TABLE1  TABLE3  TABLE3  .03,  1,  P5/3,  0
```

In this example the transmission line is connected between nodes 4, 6, and 9, and is composed of 50 LCRG sections. The values of the LCRG elements in each section are given by:

```
L = .03*TABLE2(I(L))
C = TABLE1(V(C))
R = P5/3*TABLE3(I(R))
G = 0*TABLE3(V(G)) = 0
```

where I(L) and I(R) are the currents through the series LR elements in a particular section, and V(C) and V(G) are the voltages across the shunt CG elements in a particular section. TABLE1, TABLE2, and TABLE3 are described elsewhere in the NET-2 input.

The transmission line element is implemented as a standard NET-2 circuit element using the subroutine DTLINE. An internal array is constructed for each line which represents voltages, currents, and admittance elements for each LCRG section within the line. The nonlinear behavior of the LCRG elements is calculated specifically for each element, using voltage and current dependencies as prescribed by a tabular function along with user specified coefficients.

Contributions to the internal array of voltages, currents, and admittances are made by superposition from the individual elements in the transmission line. The resultant array is transformed into a three terminal linearized Norton equivalent network. The admittance and source terms corresponding to the Norton equivalent are then inserted into the NET-2 nodal admittance matrices and the entire network is then solved in conventional fashion.

Following network solution, the terminal voltages on the transmission line are known. Through an inverse transformation, the internal voltages and currents in the transmission line element array are found. This permits the LCRG elements to be evaluated for the new voltages and currents, and the process is repeated.

Nonlinear iteration with convergence checking is used for the time domain steady state solution. The transmission line is treated as a quasi-converged system without specific convergence checking during the transient solution. The addition of iteration and convergence checking during the transient solution would appreciably increase running time and is not expected to contribute significantly to the solution accuracy during the transient. If other nonlinear elements are included in the network, the transmission line will be iterated automatically during the transient response calculation.

CHAPTER VII

POWER SERIES V-I CURVE

The PSVI element is a circuit element which may be used to represent a voltage-current characteristic curve in which the voltage across the element is represented by a power series expansion of the current flowing through the element. The voltage across the element is related to the current flowing through the element by the expression:

$$V = \sigma \sum_{i=1}^{n} K_i |I|^i$$

where: V = element voltage
I = element current
$K_1, K_2, \ldots K_n$ = power series coefficients
σ = sign of I

From the above equation it is seen that the element voltage for negative values of current is the negative of the element voltage for positive values of current, i.e., if $V = F(I)$ then $V = -F(-I)$ for $I > 0$. This means that the PSVI element behaves exactly like a nonlinear resistor whose resistance is given by:

$$R = \frac{V}{I} = \sum_{i=1}^{n} K_i |I|^{i-1}$$

The format for the element is:

PSVIn (p) a b $K_1, K_2, \ldots\ldots K_n$

where: PSVIn = element ID
p = optional parallel segment designation
a and b = node pair names
$K_1, K_2, \ldots\ldots K_n$ = power series coefficients

The power series coefficients may be represented by numbers or mathematical expressions. The element does not contain any intrinsic computational delay.

The voltage across the element is given by:

$$V = e_a - e_b$$

where e_a and e_b are the voltages at nodes a and b, respectively. The current flow through the element is given by:

$$I = V/R$$

where R is the equivalent nonlinear resistance of the element. Element power dissipation P is given by

$$P = IV$$

The user may reference the power series coefficients K_1, K_2, K_3, etc., by using the respective symbolic names PSVIn, PSVIn.1, PSVIn.2, etc. The element voltage, current and power dissipation may be referenced by using the symbolic names V(PSVIn), I(PSVIn), and P(PSVIn), respectively.

CHAPTER VIII

VOLTAGE CONTROLLED CAPACITANCE

This element models a capacitance whose value is proportional to the voltage across a node pair.

The format for the element is:

VCCn (p) f g a b K

where: VCCn = element ID
p = optional parallel segment designation
f and g = node pair node names
a and b = node names for element connection points
K = value of proportionality factor

The capacitance is inserted into the network between nodes a and b. The capacitance value C is given by

$$C = K(e_f - e_g)$$

where e_f and e_g are the node voltages at nodes f and g, respectively.

The voltage v across the capacitance is given by

$$v = e_a - e_b$$

where e_a and e_b are the node voltages at nodes a and b, respectively. The current i flowing through the capacitance is given by:

$$i = C \frac{dv}{dt}$$

The electric charge Q on the capacitance is given by

$$Q = Cv$$

The energy E stored in the electric field of the capacitance is given by

$$E = Cv^2/2$$

The user may reference the proportionality factor K, the voltage across the capacitance, the current through the capacitance, the electric charge, and the stored energy by using the symbolic names VCCn, V(VCCn), I(VCCn), Q(VCCn), and E(VCCn), respectively.